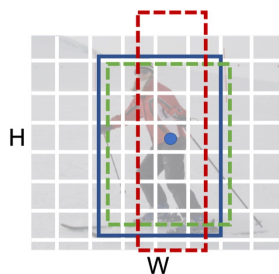
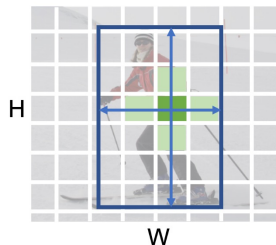

OneNet: What Makes for End-to-End Object Detection?

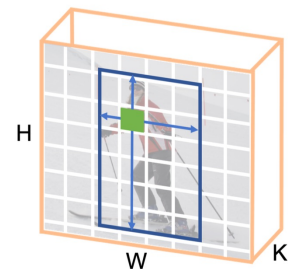
License MIT



(a) **Box Assignment:** YOLO, RetinaNet. Matching cost is only location cost of box IoU. Boxes are labeled as positive (green) if overlap IoU with ground-truth box is greater than high threshold, negative (red) if smaller than low threshold.



(b) **Point Assignment:** FCOS, CenterNet. Matching cost is only location cost of point distance. The point closest to center of ground-truth box is assigned to positive label. Nearby points have a reduced negative loss.



(c) **Minimum Cost Assignment:** OneNet. Matching cost is summation of classification cost and location cost. Positive label is assigned to only one sample of minimum cost, others are all negative ones.

Comparisons of different label assignment methods. H and W are height and width of feature map, respectively, K is number of object categories. Previous works on one-stage object detection assign labels by only position cost, such as (a) box IoU or (b) point distance between sample and ground-truth. In our method, however, (c) classification cost is additionally introduced. We discover that **classification cost is the key to the success of end-to-end**. Without classification cost, only location cost leads to redundant boxes of high confidence scores in inference, making NMS post-processing a necessary component.

Introduction

arxiv: OneNet: Towards End-to-End One-Stage Object Detection

paper: What Makes for End-to-End Object Detection?

Updates

- (28/06/2021) OneNet.RetinaNet and OneNet.FCOS on CrowdHuman are available.
- (27/06/2021) OneNet.RetinaNet and OneNet.FCOS are available.
- (11/12/2020) Higher Performance for OneNet is reported by disable gradient clip.

Comming

- ☒ Provide models and logs

- ☐ Support to caffe, onnx, tensorRT
- ☐ Support to MobileNet

Models on COCO

We provide two models - dcn is for high accuracy - nodcn is for easy deployment.

Method	inf_time	train_time	box AP	download
R18_dcn	109 FPS	20h	29.9	model log
R18_nodcn	138 FPS	13h	27.7	model log
R50_dcn	67 FPS	36h	35.7	model log
R50_nodcn	73 FPS	29h	32.7	model log
R50_RetinaNet	26 FPS	31h	37.5	model log
R50_FCOS	27 FPS	21h	38.9	model log

If download link is invalid, models and logs are also available in Github Release and Baidu Drive by code nhr8.

Notes

- We observe about 0.3 AP noise.
- The training time and inference time are on 8 NVIDIA V100 GPUs. We observe the same type of GPUs in different clusters may cost different time.
- We use the models pre-trained on imagenet using torchvision. And we provide torchvision's ResNet-18.pkl model. More details can be found in the conversion script.

Models on CrowdHuman

Method	inf_time	train_time	AP50	mMR	recall	download
R50_RetinaNet	26 FPS	11.5h	90.9	48.8	98.0	model log
R50_FCOS	27 FPS	4.5h	90.6	48.6	97.7	model log

If download link is invalid, models and logs are also available in Github Release and Baidu Drive by code nhr8.

Notes

- The evaluation code is built on top of cvpods.
- The default evaluation code in training should be ignored, since it only considers at most 100 objects in one image, while crowdhuman image contains more than 100 objects.
- The training time and inference time are on 8 NVIDIA V100 GPUs. We observe the same type of GPUs in different clusters may cost different time.
- More training steps are in the crowdhumantools.

Installation

The codebases are built on top of Detectron2 and DETR.

Requirements

- Linux or macOS with Python ≥ 3.6
- PyTorch ≥ 1.5 and torchvision that matches the PyTorch installation. You can install them together at pytorch.org to make sure of this
- OpenCV is optional and needed by demo and visualization

Steps

1. Install and build libs

```
1 git clone https://github.com/PeizeSun/OneNet.git
2 cd OneNet
3 python setup.py build develop
```

2. Link coco dataset path to OneNet/datasets/coco

```
1 mkdir -p datasets/coco
2 ln -s /path_to_coco_dataset/annotations datasets/coco/annotations
3 ln -s /path_to_coco_dataset/train2017 datasets/coco/train2017
4 ln -s /path_to_coco_dataset/val2017 datasets/coco/val2017
```

3. Train OneNet

```
1 python projects/OneNet/train_net.py --num-gpus 8 \
2     --config-file projects/OneNet/configs/onenet.res50.dcn.yaml
```

4. Evaluate OneNet

```
1 python projects/OneNet/train_net.py --num-gpus 8 \  
2   --config-file projects/OneNet/configs/onenet.res50.dcn.yaml \  
3   --eval-only MODEL.WEIGHTS path/to/model.pth
```

5. Visualize OneNet

```
1 python demo/demo.py\  
2   --config-file projects/OneNet/configs/onenet.res50.dcn.yaml \  
3   --input path/to/images --output path/to/save_images --confidence-  
4     threshold 0.4 \  
5   --opts MODEL.WEIGHTS path/to/model.pth
```

License

OneNet is released under MIT License.

Citing

If you use OneNet in your research or wish to refer to the baseline results published here, please use the following BibTeX entries:

```
1  
2 @InProceedings{peize2020onenet,  
3   title = {What Makes for End-to-End Object Detection?},  
4   author = {Sun, Peize and Jiang, Yi and Xie, Enze and Shao,  
5     Wenqi and Yuan, Zehuan and Wang, Changhu and Luo, Ping},  
6   booktitle = {Proceedings of the 38th International Conference on  
7     Machine Learning},  
8   pages = {9934--9944},  
9   year = {2021},  
10  volume = {139},  
11  series = {Proceedings of Machine Learning Research},  
12  publisher = {PMLR},  
13 }
```